Millipede cluster user guide

Fokke Dijkstra

HPC/V

Donald Smits Centre for Information Technology

May 2010

1 Introduction

This user guide has been written to help new users of the Millipede HPC cluster at the CIT getting started in using the cluster.

1.1 Common notation

Commands that can be typed in on the Linux command line are denoted with:

```
$ command
```

The `$` sign is what Linux will present you with after logging in. After this `$` sign you can give commands, denoted with `command` above, which you can just input using the keyboard. You have to confirm the command with the <Enter> key.

1.2 Cluster setup

1.2.1 Hardware setup

The Millipede cluster is a heterogeneous cluster consisting of 4 parts.

1. Two front-end nodes where users login to with 4 3GHzAMD Opteron cores, 8 GB of memory and 7TB of diskspace;
2. 236 nodes with 12 2.6 GHz AMD Opteron cores, 24GB of memory, and 320 GB of local disk space;
3. 16 nodes with 24 2.6 GHz AMD Opteron cores, 128 GB of memory, and 320 GB of local disk space;
4. 1 node with 64 cores and 512 GB of memory.

All the nodes are connected with a 20Gbps Infiniband network. Attached to the cluster is 110TB of storage space, which is accessible from all the nodes.

(To get some idea of the power of the machines, a normal desktop PC now has 2 cores running at 2.6 GHz, 4 GB of memory and 1 TB of diskspace.)

1.2.2 Login node

Two of the nodes of the cluster are used as a login node. These are the nodes you login to with the username and password given to you by the system administrator. The other nodes in the cluster are so called 'batch' nodes. They are used to perform calculations on behalf of the users. These nodes can only be reached through the job scheduler. In order to use these a description of what you want the node(s) to do has to be written first. This description is called a job. How to submit jobs will be

explained later on.

1.2.3 File systems

The cluster has a number of file systems that can be used. On Unix systems these file systems are not pointed to with a drive letter, like on Windows systems but appear as a certain directory path. The file systems available on the system are:

/home

This file system is the place where you arrive after logging in to the system. Every user has a private directory on this file system. Your directory on /home, and its subdirectories are available on all the nodes of the system.
You can use this directory to store your programs and data. In order to prevent the system from running out of space the amount of data you can store here is limited, however. On the /home file system quota are in place to prevent a user from filling up all the available disk space. This means that you can only store a limited amount of data on the file system. For /home the amount of space is limited to 10 GB. When you are in need of more space you should contact the system administrators to discuss this, and depending on your requirements and the availability your quota may be changed.
The data stored on /home is backed up every night to prevent data loss in case the file system breaks down or because of user or administrative errors. If you need data to be restored you can ask the site administrators to do this, but of course it is better to be careful when removing data.
Note, however, that using the home directory for reading or writing large amounts of data may be slow. In some cases it may be useful to copy input data from your home directory to `/data/scratch/$TMPDIR` on the batch node first at the beginning of your job. Note that relevant output has to be copied back at the end of the job, otherwise it will be lost, because `/data/scratch/$TMPDIR` is automatically cleaned up after your job finishes.

/data

For storing large data sets a file system /data has been created. This file system is 110 TB large. Part of it is meant for temporary usage (/data/scratch), the rest is for permanent storage. In order to prevent the file system from running out of space there is a limit to how much you can store on the file system. The current limit is 200 GB per user. There is no active quota system, but when you use more space you will be sent a reminder to clean up.

The /data file system is a fast clustered file system that is well suited for storing large data sets. Because of the amount of disk space involved no backup is done on these files, however.

/data/scratch

The file system mounted at /scratch is a temporary space that can be used by your jobs while they are running. For each job a temporary directory is located. This directory can be reached through the environment variable $TMPDIR. This space is automatically cleaned up after your job is finished. Note that relevant output has therefore to be copied back at the end of the job, otherwise it will be lost.
Files you store on /data/scratch at other locations will be removed after a couple of days.
In some cases it may be useful to copy input data from your home directory to the temporary directory on /data/scratch at the beginning of your job. This because the /home file system is not very fast.

1.3 Prerequisites for cluster jobs

Programs that need to be run on the cluster need to fulfil some requirements. These are:

1. The program should be able to run under Linux. If in doubt, the author of the program should be able to help you with this. Some hints:
    a. It is helpful if there is source code available so that you can compile the program yourself.
    b. Programs written in Fortran, C or C++ can in principle be compiled on the cluster
    c. Java programs can also be run because Java is platform independent.
    d. Some scripting languages like e.g. Python or Perl can also be used
2. Programs running on the batch nodes can not easily be run interactively. This means that it is in principle not possible to run programs that expect input from you while they are running. This makes it hard to run programs that use a graphical user interface (GUI) for controlling them. Note also that jobs may run in the middle of the night or during the weekend, so it is also much easier for you if you don't have to interfere with the jobs while they are running.
   It is possible, however, to startup interactive jobs. These are still scheduled, but you will be presented with a command line prompt when they are started.
3. Matlab and R are also available on the cluster and can be run in batch mode (where the graphical user interface is not displayed).

If you have any questions on how to run your programs on the cluster, please contact the CIT central service desk.

2 Obtaining an account

The Millipede system is available to support scientific research and education. University staff members that want to use the system for these purposes can request an account. Students may also use the system for these purposes, but will need the approval of a staff member for this. The accounts can therefore only be requested by staff members.
People not affiliated to the University of Groningen can only get an account under special circumstances. Please contact the CIT central Service Desk if you want more information on this.
In order to get an account on the system you will have to answer the following questions.

Requestor
Full name:
Registration number (p-number):
Affiliation:
Description of the intended use of the account (few sentences, at most half a page of A4). This information is mainly for the CIT to get some idea about what the cluster is actually used for.
Telephone number:
E-mail address:

User (if different from the requestor, e.g. in the case of a student)
Full name:
Registration number (p- or s-number):
Telephone number:

E-mail address:

This information can be sent to the CIT central Service Desk. When an account has been created the user will be contacted about the user name and password.

3 Logging in

Since the login procedure for Windows users is rather different from that for Linux users we will describe these in different sections. Logging in from Mac OS X is also possible using the Terminal, but this is not further described here.

If you need assistance with logging into the system, please contact the CIT central service desk.

3.1 Windows users

3.1.1 Available software

Windows users will need to install SSH client software in order to be able to login into the cluster. The following clients are useful:

PuTTY+ WinSCP

PuTTY is a free open source SSH client. PuTTY is available on the standard RUG Windows desktop. If you are not using this, PuTTY can be downloaded from
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
For most users, getting and installing the installer version is the easiest.

A free open source file transfer utility is WinSCP. This utility is also available on the standard RUG Windows desktop. It can be downloaded from:
http://winscp.net/eng/index.php

Optional: X-server

For displaying programs with a Graphical User Interface (GUI) an X server is needed. A free open source X server for Windows is Xming. Xming can be downloaded from:
http://sourceforge.net/projects/xming

3.1.2 Using the software

PuTTY

When starting up PuTTY you will be confronted with the screen in Figure 1, where you can enter the name of the machine you want to connect to.

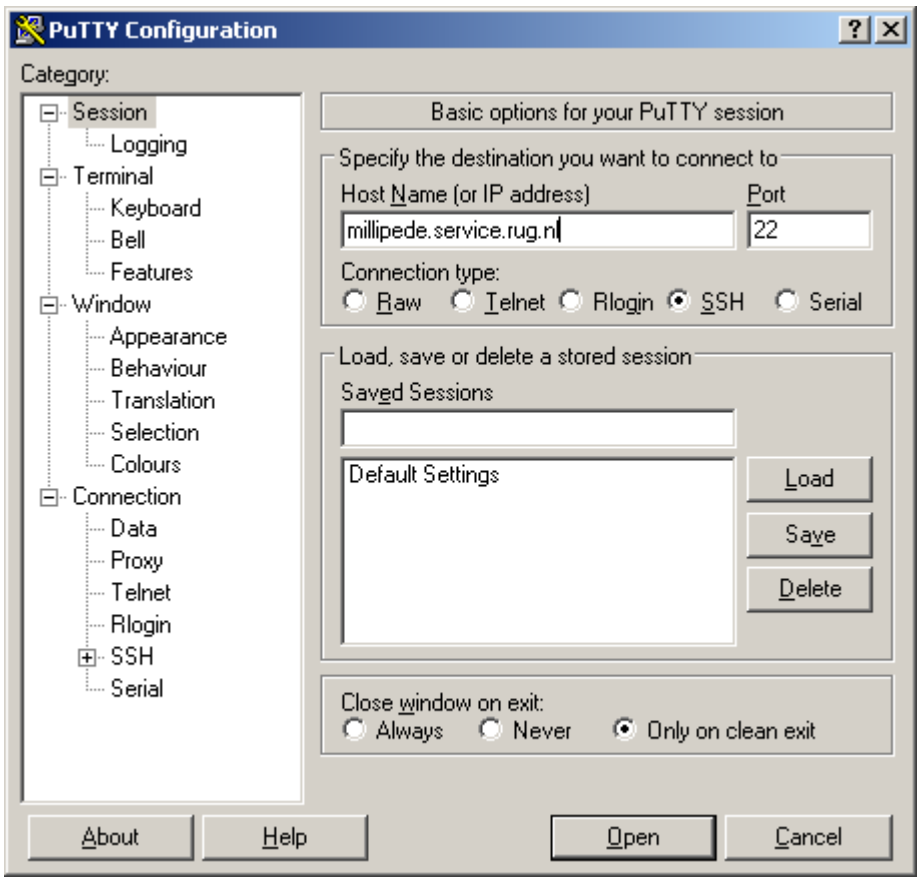

Figure 1 - PuTTY startup screen

For logging in into Millipede you should use the hostname millipede.service.rug.nl as shown in the Figure. You can confirm your input by clicking on “Open”. Note that the port number “22” does not have to be changed.

When connecting to a machine for the first time its host key will not be known. PuTTY will therefore ask you if you trust the machine and if you want to store its host key (Figure 2). When connecting for the first time just say “Yes” here. This will store the host key in PuTTY's cache and you should not see this dialog again for the machine you want to connect to.

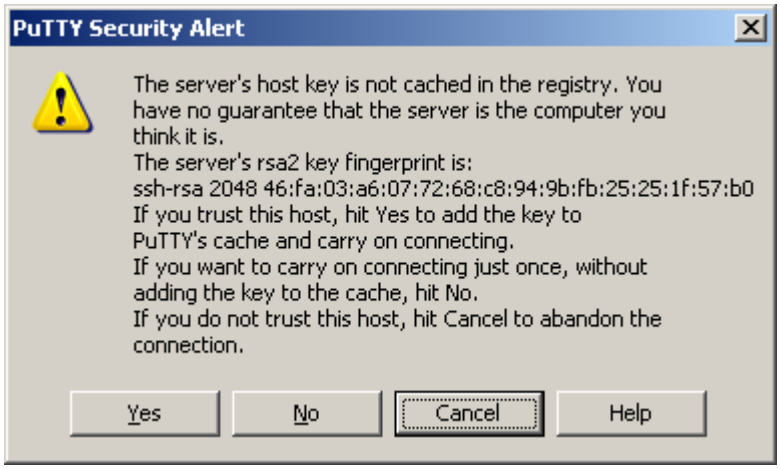
PuTTY Security Alert
The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 46:fa:03:a6:07:72:68:c8:94:9b:fb:25:25:1f:57:b0
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.
Yes
No
Cancel
Help

Figure 2 - Save host key dialog

After connecting PuTTY will open a terminal window (Figure 3). Here you first will have to enter your username, followed by the password. You should have obtained the username and password

from the CIT Service Desk when applying for an account.

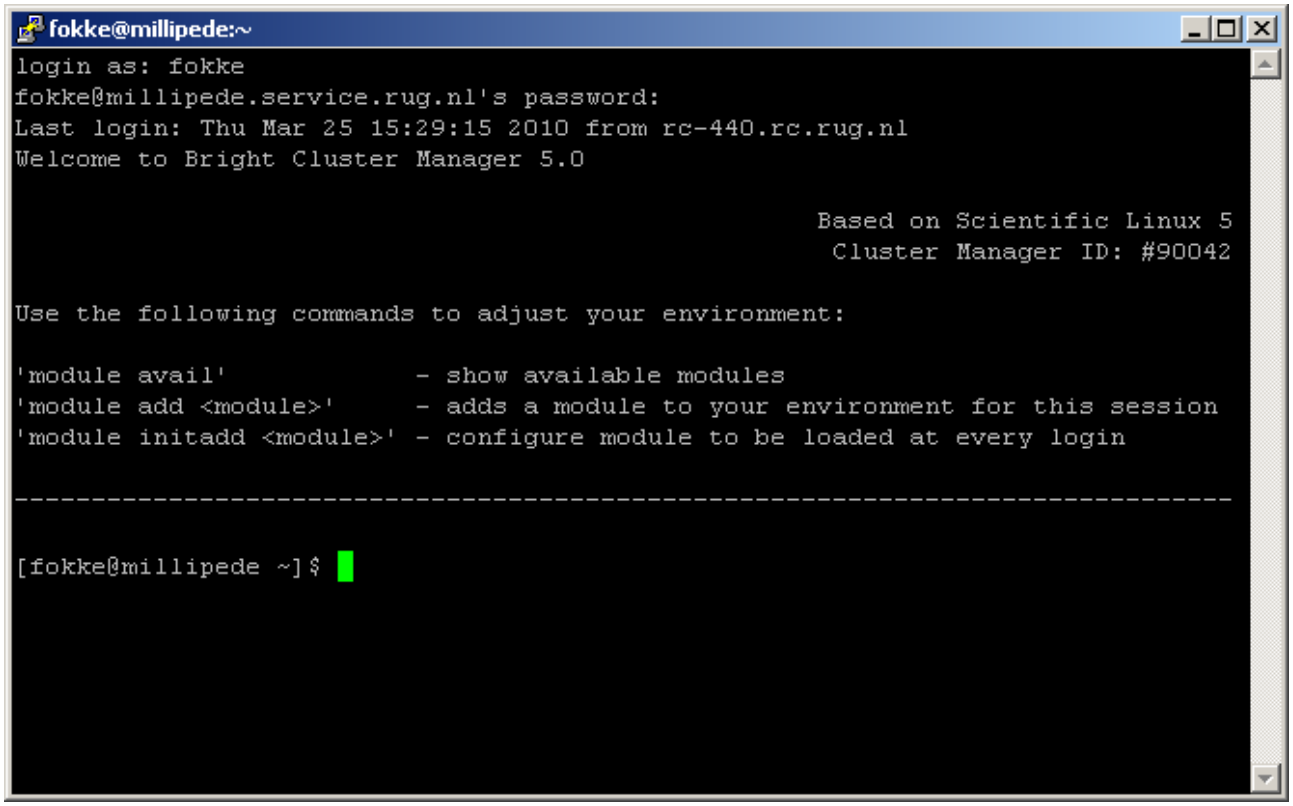
fokke@millipede:~
login as: fokke
fokke@millipede.service.rug.nl's password:
Last login: Thu Mar 25 15:29:15 2010 from rc-440.rc.rug.nl
Welcome to Bright Cluster Manager 5.0
Based on Scientific Linux 5
Cluster Manager ID: #90042
Use the following commands to adjust your environment:
'module avail' - show available modules
'module add <module>' - adds a module to your environment for this session
'module initadd <module>' - configure module to be loaded at every login
[fokke@millipede ~]$

Figure 3 - Putty terminal window

In order to prevent some work when logging in the next time it is possible to save your session. This will store the preferences you made when connecting in a session, which can be used to easily reconnect later on. In order to save your session (see Figure 4) you have to enter a name in the "Saved sessions" box at the PuTTY startup screen. If you then click "Save" the settings you have supplied like "host name" will be saved in a session with the given name. You can of course also change the "Default Settings" session.

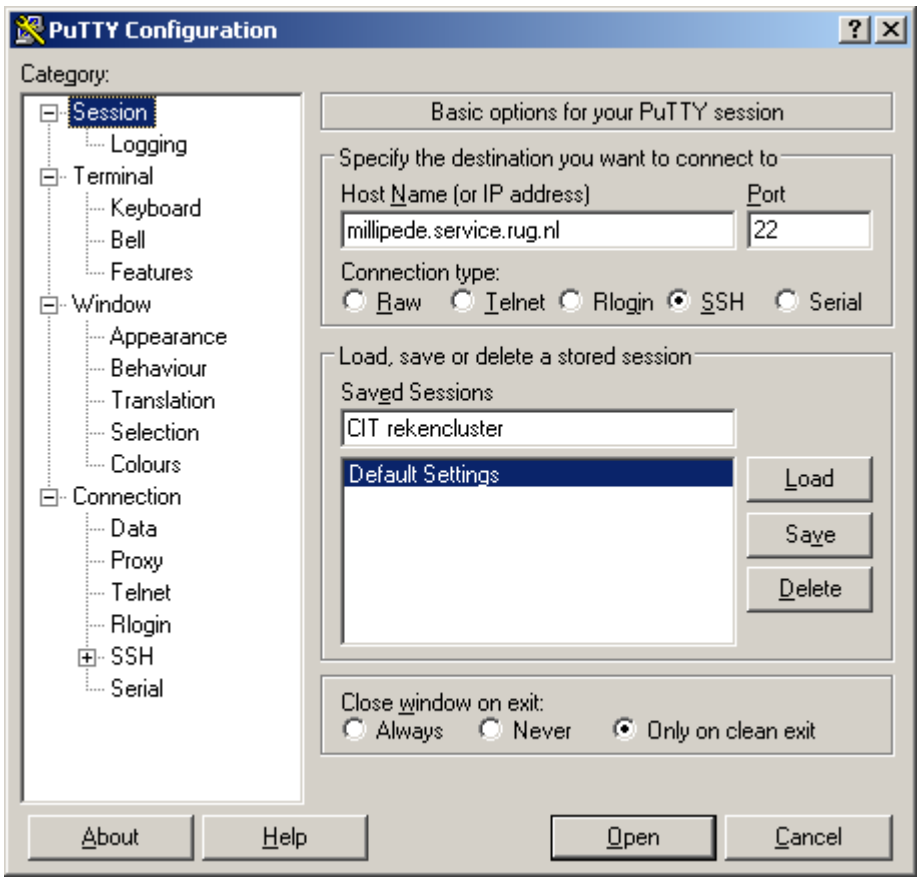

Figure 4 - Save session dialog

Supplying a standard username is one of the things that can be very useful, especially when saved in a session. The username can be supplied when selecting Connection→Data in the left side of the window (Figure 5).

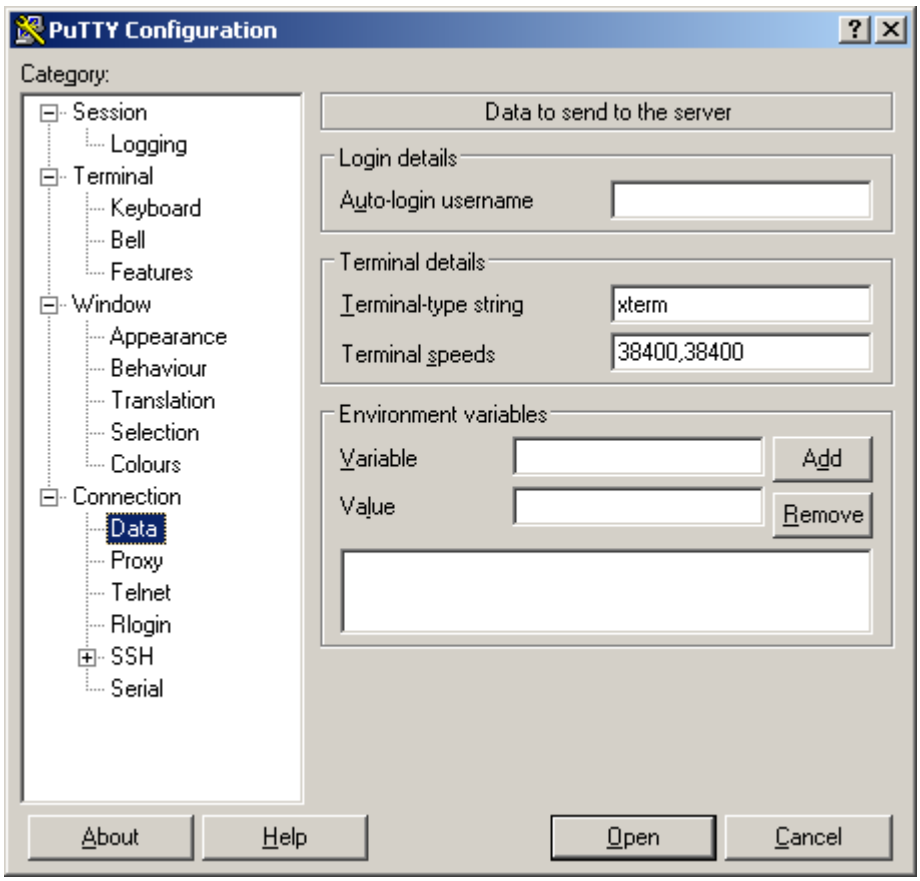

Figure 5 - Supplying a standard username

Some programs may want to show graphical output in e.g. a graphical user interface (GUI). On Unix systems the X11 protocol is commonly used to draw graphics. These graphics can be displayed on remote systems like your desktop machine. For this an X-server program needs to run on your desktop (see the section on Xming, further on for more details). Normally X11 traffic goes unsecured over the network. This can lead to various security problems. Furthermore network ports would need to be opened up on your machine. Fortunately tunneling X11 connections over ssh solves this problem, it also makes displaying programs easier because no further setup is necessary. In order to enable tunneled X11 connections the checkbox “Enable X11 forwarding” shown in Figure 6 has to be set. This checkbox can be found at Connection→SSH→X11 in the left hand side of the window. Note that it is easiest to save this in a profile so that it is always on.

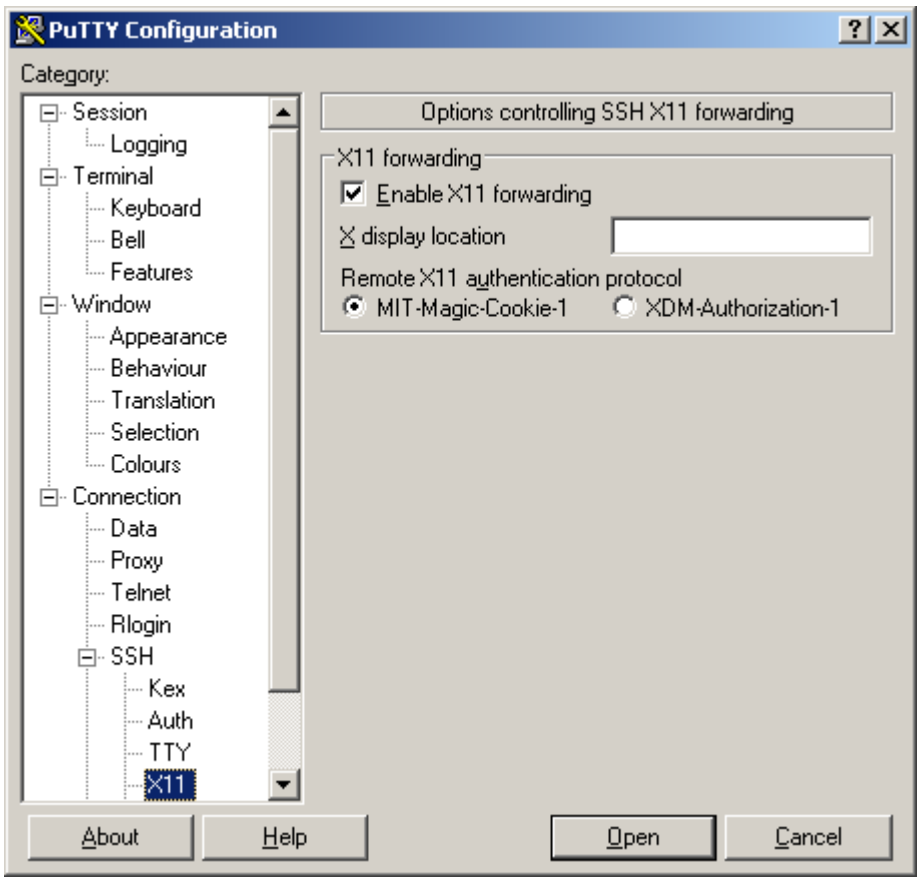

Figure 6 - Enable X11 forwarding to be able to display graphics

WinSCP

When starting up the file transfer client WinSCP, you will be presented with the screen shown in Figure 7. You will have to enter the machine name, username and password here in order to make a connection to the remote system. Is is also possible to save this input into a session. Note that you should NOT save your password into these sessions!

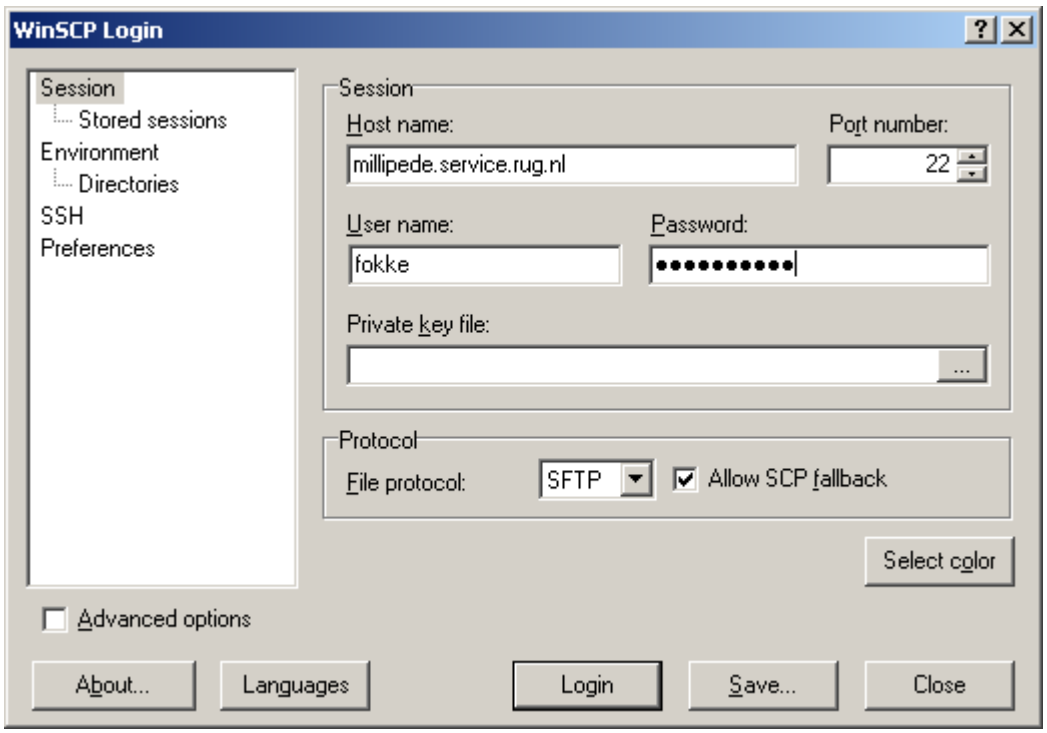

Figure 7 - WinSCP login screen

When connecting to a machine for the first time the host key of this machine will not be known to WinSCP. It will therefore offer to store the key into its cache. You can safely press “Yes” here (Figure 8).

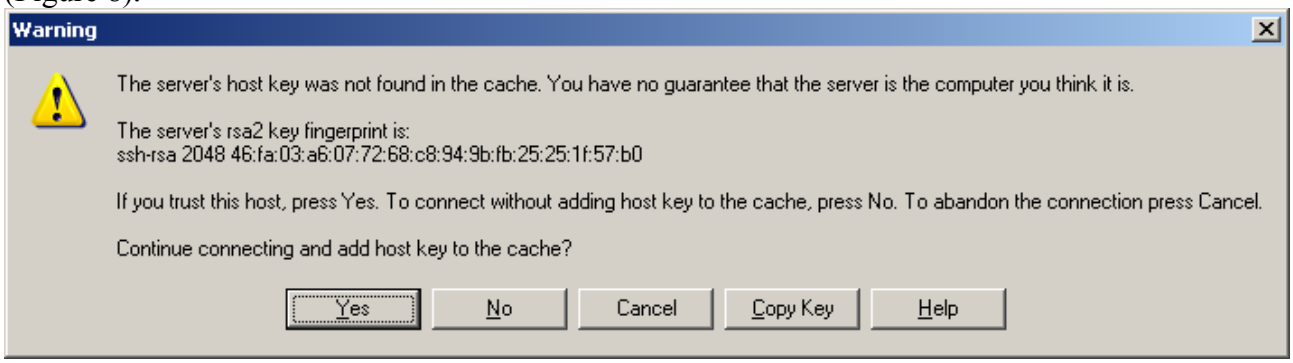
Warning
The server's host key was not found in the cache. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 46:fa:03:a6:07:72:68:c8:94:9b:fb:25:25:1f:57:b0
If you trust this host, press Yes. To connect without adding host key to the cache, press No. To abandon the connection press Cancel.
Continue connecting and add host key to the cache?
Yes
No
Cancel
Copy Key
Help

Figure 8 - WinSCP save hostkey dialog

After the connection has been made you will be presented with the file transfer screen (Figure 9). It will show a local directory on the left side and a directory on the remote machine on the right. You can transfer files by dragging them from left to right or vice-versa. The current directory can be changed by making use of the icons above the directory info screens.

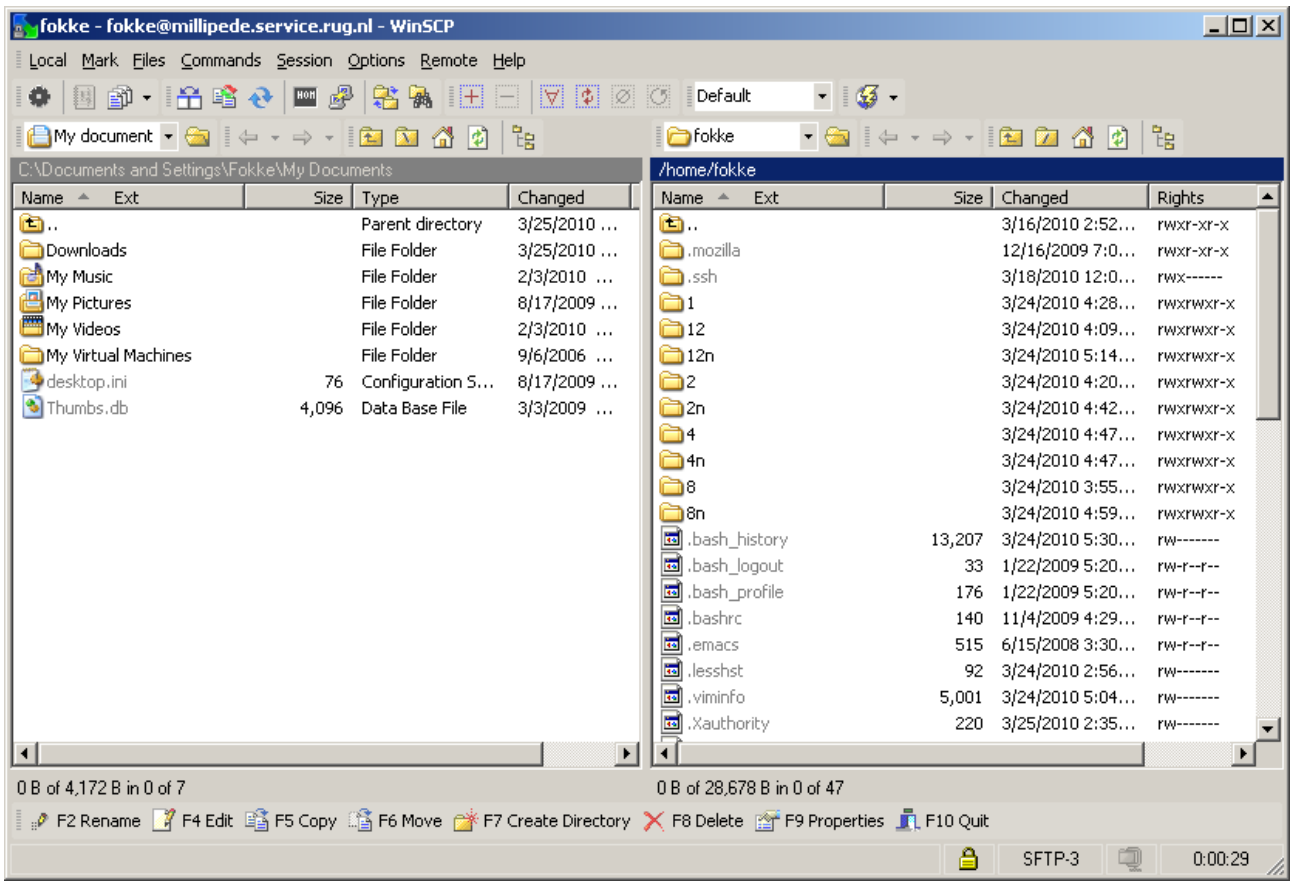
fokke - fokke@millipede.service.rug.nl - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
My document
fokke
C:\Documents and Settings\Fokke\My Documents
/home/fokke
Name Ext Size Type Changed
Name Ext Size Changed Rights
0 B of 4,172 B in 0 of 7
0 B of 28,678 B in 0 of 47
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
0:00:29

Figure 9 - WinSCP file transfer window

Xming

To be able to run programs on the cluster that display some graphical output, an X-server must be running on your local desktop machine. Xming is such an X-server and it is open source and freely available.
When starting Xming for the first time, Windows may ask you if it should allow Xming to accept connections from the outside (Figure 10). Since you should always use tunneled X11 connections Xming does not have to be reachable from the outside. So answer “Keep blocking” when presented with this dialog.

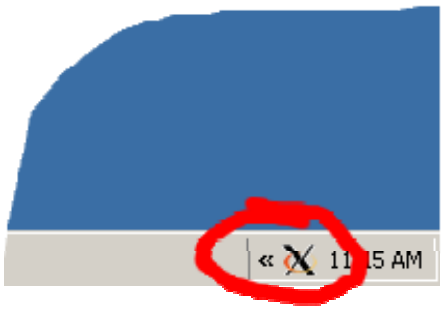

Figure 10 - Xming traybar icon

When Xming is running it will be able to show graphical displays of programs running on the cluster. This under the provision that you have enabled X11 Forwarding on your SSH client (like PuTTY). When running, Xming will show an icon with an X in the system tray (Figure 9).

Note that transferring this graphical data requires some bandwidth. It is therefore only really usable when connected to the university network directly. When using this at home you may notice that the drawing of the windows is very slow.

Note that the following problem described by Chieh Cheng (http://www.gearhack.com/Forums/DisplayComments.php?file=Computer/Linux/Troubleshooting._X_connection_to_localhost.10.0_broken_.explicit_kill_or_server_shutdown..) exists when running Xming under Windows Vista. For Xming to work correctly the localhost entry must be available in the hosts file (%SYSTEMROOT%\System32\drivers\etc\hosts, where %SYSTEMROOT% is normally C:\Windows). It must contain the entry "127.0.0.1 localhost". On Vista it only contains the entry "::1 localhost", which is for IPv6 instead of IPv4. When the correct entry is not present, you will get "X connection to localhost:10.0 broken (explicit kill or server shutdown)" errors when you try to launch an X client application.

3.2 Linux users

For Linux distributions all necessary software should already be included. A connection to the cluster can be made from a terminal window. The command to login is then:

```
$ ssh -X username@millipede.service.rug.nl
```

Here `username` should be replaced by your username. After that you should give your password. The "-X" option will enable X11 Forwarding, which is necessary to be able to display graphical output from programs running on the cluster. Note that this option may be the default setting on your system.

4 Working with Linux

4.1 The Linux command line prompt

After logging in you will be presented with a command prompt. Here you can enter commands for the login node. A nice introduction to using the Linux command prompt can be found at: http://www.linuxcommand.org/

Since this webpage already contains a nice tutorial on how to use the command line this information will not be copied here.

More information on Linux can also be found on the following websites:

- Machtelt Garrels, Introduction to Linux: http://tille.garrels.be/training/tldp/
- Scott Morris, The easiest Linux guide you'll ever read: http://www.suseblog.com/my-book-the-easiest-linux-guide-youll-ever-read-an-introduction-to-linux-for-windows-users

4.2 Editors

On the system several editors are available, including emacs and vi. For beginners nano is probably the easiest to use.

4.2.1 Nano

Editing text files is often necessary to create or change for example input files or job scripts. The easiest editor available on the HPC cluster is nano. You can start nano by issuing the following

command:
```
$ nano
```
You can also start editing a file by issuing:
```
$ nano filename
```

When nano is started you will be presented with the screen shown in Figure 11.

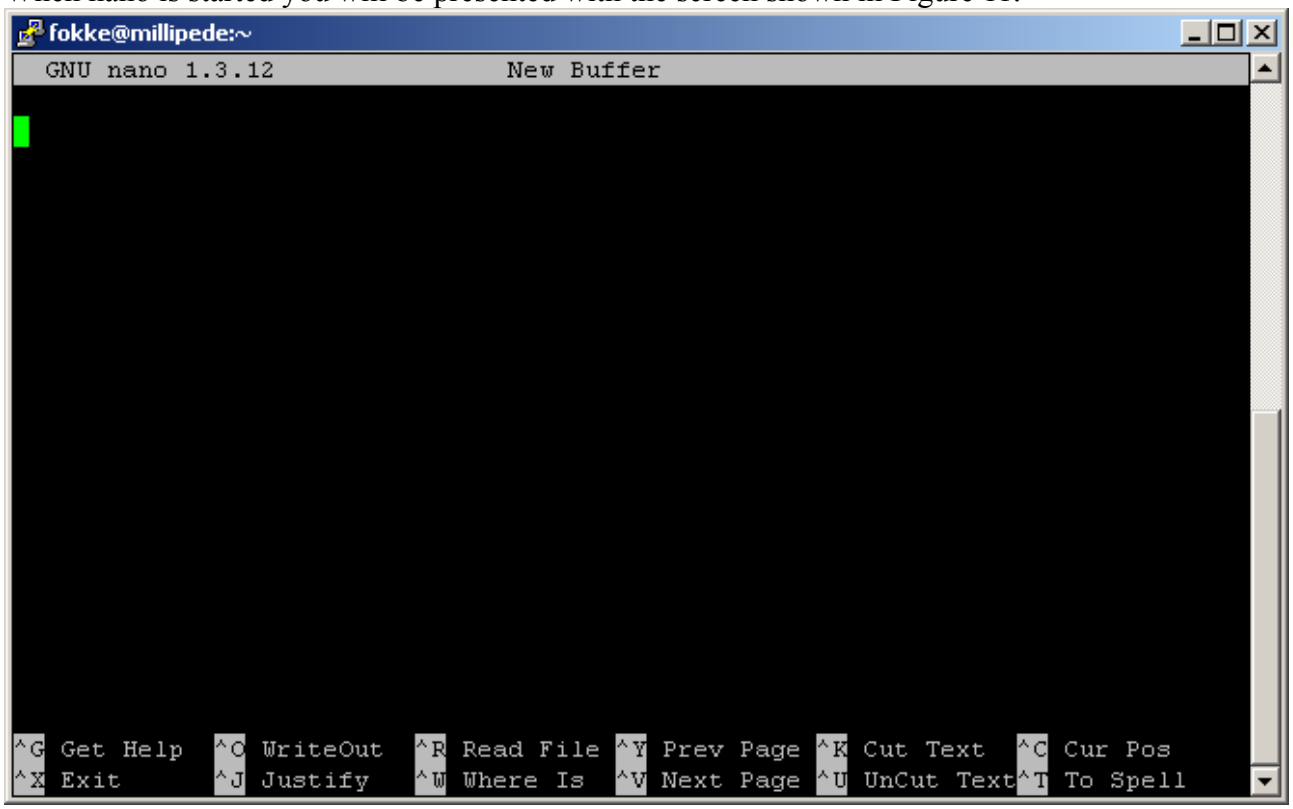
fokke@millipede:~
GNU nano 1.3.12 New Buffer
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Figure 11 - Nano editor

You can add text by simply typing what you want. The table at the bottom of the screen shows the commands that can be given to quit, save your text etc. These commands can be accessed by using <Ctrl>, denoted with ^, together with the key given. <Ctrl>-X will for example quit the editor. <Ctrl>-O will save the current text to file, etc.

4.2.2 Using WinSCP

Another probably easier way to edit files is to use WinSCP. When double-clicking on a file stored on the cluster in WinSCP an editor will be fired up. When you save your changes the changed file will be transferred back to the cluster.

4.2.3 End of line difference between Windows and Linux

A small problem you may run into is that there is a difference between Linux and Windows in the way “end of line” is represented. Windows represents the “end of line” by two characters, namely “carriage return” and “linefeed” (CRLF), where Linux uses a single “linefeed”. When editing a file created on Linux with e.g. Notepad on Windows, the file may appear as a single line of text with □ characters where the line breaks should be. A file created on Windows may appear to have extra “^M” characters at the line break positions on Linux systems.

Many current applications do not have problems recognizing the different form of “end of line”

however. The WinSCP editor can handle both file types. When problems appear at the Linux side opening and saving the file with “nano” will solve the problem. Note, however, that most shell interpreters like `bash` or `csh` will have problems when the wrong “end of line” characters are used.

A file with the Windows CRLF end of line can be detected on Linux by using the command “file”. A Linux text file will result in the following output:

```
$ file testfile
testfile: ASCII text
```

A Windows text file will give the following:

```
$ file testfile
testfile: ASCII text, with CRLF line terminators
```

This does not work for shell scripts however. In this case the cat command can be used instead. When `cat` is used with the option `-v`, the file is shown as is, including the CR characters. This will result in `^M` being displayed at the end of each line:

```
$ cat -v testfile
This is a textfile created on a MS Windows system^M
It has CRLF as linefeed^M
This may give problems on Linux systems^M
```

5 Module environment

On the system a wide variety of software is available for you to use. In order to make life easier for the users, the module system has been installed to help you setting up the correct environment for the different software packages. This also allows the user to select a specific version of a software package.

5.1 Module command

The environment can be set using the “module” command. Some useful available options for the command are:

`avail`	List the available software modules
`list`	List the modules you have currently loaded into your environment
`add <module name>`	Add a module to your environment
`rm <module name>`	Remove a module from your environment
`purge`	Remove all modules from your environment
`initadd <module name>`	Add a module to your initial environment, so that it will always be loaded.
`initrm <module name>`	Remove a module from your initial environment.
`whatis <module name>`	Gives an explanation of what software a certain module is for.

5.2 Using the command

To see the available module in the system you can use the “avail” command like:

```
$ module avail

---------------------------- /cm/local/modulefiles -----------------------------
3ware/9.5.2       dot               null              version
cluster-tools/5.0 freeipmi/0.7.11   openldap
cmd               ipmitool/1.8.11   shared
cmsh              module-info       use.own

---------------------------- /cm/shared/modulefiles ----------------------------
R/2.10.1                          intel/compiler/32/11.1/046
acml/gcc/64/4.3.0                 intel/compiler/64/11.1/046
acml/gcc/mp/64/4.3.0              intel-cluster-checker/1.3
acml/gcc-int64/64/4.3.0           intel-cluster-runtime/2.1
acml/gcc-int64/mp/64/4.3.0        intel-tbb/ia32/22_20090809oss
acml/open64/64/4.3.0              intel-tbb/intel64/22_20090809oss
....
```

To show the modules currently loaded into your environment you can use the "list" command, like

```
$ module list
Currently Loaded Modulefiles:
  1) gcc/4.3.4       2) maui/3.2.6p21   3) torque/2.3.7
```

To add a module (or multiple modules) to your environment you can use the "add" command:

```
$ module add intel/compiler/64 openmpi/intel
```

When you want to load a module each time you login you can use the initadd command:

```
$ module initadd intel/compiler/64
```

6 Available software (It is hard to give general advices here????)

Several software packages have been preinstalled on the system. For most people it should be clear what packages they want to use, because their program depends on it. With respect to compilers and some numerical libraries this can be more difficult, because they offer the same functionality.

6.1 Compilers

The following compilers are available on the system:

- GNU compilers. Standard compiler suite on Linux systems.
- Intel compilers. High performance compiler developed by Intel.
- Open64 compilers. Compiler suite recommended by AMD (http://blogs.amd.com/nigeldessau/tag/open64/)
- Pathscale compilers.

6.2 MPI libraries

Several MPI libraries are available on the system:

- LAM. LAM MPI implementation. Officially superseded by OpenMPI
- MPICH. MPI-1 implementation.
- MPICH2. Implementation of MPI-1 and MPI-2
- MVAPICH. MPI-1 implementation using the Infiniband interconnect
- MVAPICH2. MPI-1 and MPI-2 implementation using Infiniband interconnect
- OpenMPI. OpenMPI MPI implementation of MPI-1 and MPI-2, supports both Infiniband and the torque scheduler for starting processes.

Since the cluster is equipped with an Infiniband interconnect the MVAPICH2 and OpenMPI implementations are the two recommended ones to use.

Note that there are versions specific to the different compilers installed. The following command will load OpenMPI for the intel compiler into your environment:

```
$ module add openmpi/intel
```

7 Submitting jobs

The login nodes of the cluster should only be used for editing files, compiling programs and very small tests (about a minute). If you perform large calculations on the login node you will hinder other people in their work. Furthermore you are limited to that single node and might therefore as well run the calculation on your desktop machine.

In order to perform larger calculations you will have to run your work on one or more of the so called ‘batch’ nodes. These nodes can only be reached through a workload management system. The task of the workload management system is to allocate resources (like processor cores and memory) to the jobs of the cluster users. Only one job can make use of a given core and a piece of memory at a time. When all cores are occupied no new jobs can be started and these will have to wait and are placed in a queue. The workload management system fulfils tasks like monitoring the compute nodes in the system, controlling the jobs (starting and stopping them), and monitoring job status.

The priority in the queue depends on the cluster usage of the user in the recent past. Each user has a share of the cluster. When the user has not been using that share in the recent past his priority for new jobs will be high. When the user has been doing a lot of work, and has gone above his share, his priority will decrease. In this way no single user can use the whole cluster for a long period of time, preventing other users from doing their work. It also allows users to submit a lot of jobs in a short period of time, without having to worry about the effect that may have on other users of the system.

The workload management and scheduling system used on the cluster is the combination of torque for the workload management and maui for the scheduling. More information about this software can be found at http://www.clusterresources.com/

Note that you may have to add torque and maui to your environment first, before you can use the commands described below. You can do this using:

```
$ module add torque maui
```

7.1 Job script

In order to run a job on the cluster a job script should be constructed first. This script contains the commands that you want to run. It also contains special lines starting with "#PBS". These lines are interpreted by the torque workload management system. An example is given below:

```
#!/bin/bash
#PBS -N myjob
#PBS -l nodes=1:ppn=2
#PBS –l mem=500mb
#PBS -l walltime=02:00:00

cd my_work_directory
myprogram a b c
```

Here is a description of what it does:

`#!/bin/bash`	The interpreter used to run the script if run directly. `/bin/bash` in this case.

The lines starting with #PBS are instructions for the job scheduler on the system.

`#PBS -N myjob`	This is used to attach a name to the job. This name will be displayed in the status listings.
`#PBS -l nodes=1:ppn=2`	Request 2 cores (ppn=2) on 1 computer (nodes).
`#PBS –l mem=500mb`	Request 500 MB of memory for the job.
`#PBS -l walltime=02:00:00`	The job may take at most 2 hours. The format is hours:minutes:seconds. After this time has passed the job will be removed from the system, even when it was not finished! So please be sure to select enough time here. Note, however that giving much more time than necessary may lead to a longer waiting time in the queue when the scheduler is unable to find a free spot.
`cd my_work_directory`	Go to the directory where my input files are
`myprog a b c`	Start my program called myprog with the parameters a b and c.

7.2 Submitting the job

The job script can be submitted to the scheduler using the `qsub` command, where `job_script` is the name of the script to submit:

```
$ qsub job_script
1421463.master
```

The command returns with the id of the submitted job. In principle you do not have to remember this id as it can be easily retrieved later on.

7.3 Checking job status

The status of the job can be requested using the commands `qstat` or `showq`. The difference between the commands is that `showq` shows jobs in order of remaining time when jobs are running or priority when jobs are still scheduled, while `qstat` will show the jobs in order of appearance in the system (by job id).

Here are some examples:

```
$ qstat
Job id            Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
1415138.master   dopc-ves         karel            00:00:00 R nodes
1416095.master   run_16384_obj    isabel           00:01:07 R nodes
1417470.master   ZyPos            jan              00:01:01 R quads
1417471.master   ZyPos            jan              00:01:01 R quads
1419870.master   dopc-ves         karel            00:00:00 R nodes
1420331.master   CLOSED-cAMP-4    klaske           00:00:00 R nodes
1420332.master   CLOSED-APO-4     klaske           00:00:00 R nodes
1420371.master   BUTMON           bill             00:00:00 R nodes
1420378.master   LACRIP2          klaske           00:00:00 R nodes
1420406.master   tension-14       lara             00:00:00 R smp
1420409.master   BUTMON           pieter           00:00:00 R nodes
1420413.master   Celiac4          graham           00:00:08 R nodes
1420414.master   job100           william          00:00:00 R nodes
1420415.master   But200           william          00:00:00 R nodes
1420417.master   quad-tension-7   lara             00:00:00 R smp
1420419.master   DPPC-try9        john             00:00:00 R smp
1420420.master   OPEN-APO-6       klaske           00:00:00 R nodes
....
....

$ showq
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME

1421394             william    Running     1    00:08:32  Tue May  6 14:44:36
1421395             william    Running     1    00:08:38  Tue May  6 14:44:42
1421396             william    Running     1    00:08:42  Tue May  6 14:44:46
1420406                lara    Running     4    00:25:19  Mon May  5 15:11:23
1420331              klaske    Running     2     1:15:57  Sun May  4 16:02:01
1420332              klaske    Running     2     1:19:50  Sun May  4 16:05:54
1420417                lara    Running     4     2:37:57  Mon May  5 17:24:01
1420419                john    Running    12     3:22:29  Mon May  5 18:08:33
1420423              thomas    Running    24    17:53:12  Tue May  6 08:39:16
...
...
1420509                lara    Running    16  3:19:47:48  Tue May  6 10:33:52
1419870               karel    Running     4  5:22:44:49  Fri May  2 13:30:53
1420413             graham     Running     2  9:01:25:58  Mon May  5 16:12:02

   144 Active Jobs     394 of  394 Processors Active (100.00%)
                       197 of  197 Nodes Active      (100.00%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

1420672              thomas       Idle     1  1:00:00:00  Tue May  6 11:11:25
1420673              thomas       Idle     1  1:00:00:00  Tue May  6 11:11:25
1420674              thomas       Idle     1  1:00:00:00  Tue May  6 11:11:26
...
...
```

A useful option for both commands is the `-u` option which will only show jobs for the given user, e.g.

```
$ showq -u peter
```

will only show the jobs of user peter.

It may also be useful to use less to list the output per page. This can be done by *piping* the output to less using |. (This symbol can on US-international keyboards be found close to the <Enter> key. "<Shift>\")

```
$ showq | less
```

The result of the command will be displayed per page. <PgUp> and <PgDn> can be used to scroll through the text, as well as the up and down arrow. Pressing q will exit less.

7.4 Cancelling jobs

If you discover that a job is or will not be running as it should you can remove the job from the queuing system using the `qdel` command.

```
$ qdel jobid
```

Here `jobid` is the id of the job you want to cancel. You can easily find the ids of your jobs by using `qstat` or `showq`.

7.5 Queues

Because the cluster has three types of nodes available for jobs queues have been created that match these nodes. These queues are:

- nodes: Containing the 12 core nodes with 24 GB of memory
- quads: Containing the 24 core nodes with 128 GB of memory
- smp: Containing the single 64 core node with 1TB of memory

These three queues have a maximum wallclock time limit of 1 day. The default limit for a job is only 2 hours, which means that you have to set the correct limit yourself. Using a good estimate will improve the scheduling of your jobs.

For longer running jobs two long versions of these queues have been created as well. These queues are limited to use at most half of the system though. These queues have the suffix long after the node type. On the smp node there is no long queue, because we want to prevent a single user from using the system for too long, blocking the other users. The two long queues are therefore nodeslong and quadslong.

Furthermore two special queues are available

- short: Queue for small testjobs that run for no longer than 30 minutes. These jobs will be started quickly, because some nodes have been reserved for these jobs. This queue is only available on the normal 12 core nodes.
- md: Queue for the molecular dynamics group for running jobs on their own share of the system

The default queue you will be put into when submitting a job is the "nodes" queue. If you want to use a different type of machine, you will have to select the queue for these machines explicitly. This can be done using the -q option on the commandline:

```
$ qsub –q smp myjob
```

7.6 Parallel jobs

There are several ways to run parallel jobs that use more than a single core. They can be grouped in two main flavours. Jobs that use a shared memory programming model, and those that use a distributed memory programming model. Since the first depend on shared memory between the cores these can only be run on a single node. The latter are able to run using multiple nodes.

7.6.1 Shared memory jobs

Jobs that need shared memory can only run on a single node. Because there are three types of nodes the amount of cores that you want to use and the amount of memory that you need, determine the nodes that are available for your job.

For obtaining a set of cores on a single node you will need the PBS directive:

#PBS –l nodes=1:ppn=*n*

where you have to replace *n* by the number of cores that you want to use. You will later have to submit to the queue of the node type that you want to use.

7.6.2 Distributed memory jobs

Jobs that do not depend on shared memory can run on more than a single node. This leads to a job requirement for nodes that looks like:

#PBS –l nodes=*n*:ppn=*m*

Where *n* is the number of nodes (computers) that you want to use and *m* is the number of cores per computer that you want to use. If you want to use full nodes the number *m* should be equal to the number of cores per node.

7.7 Memory requirements

By default a job will have a memory requirement per process that is equal to the available memory of a node divided by the number of cores. This means that for each process in your job this amount is available. If you need more (or less) than this amount of memory, you should specify this in you job requirement by adding a line:

#PBS –l pmem=*x*G

This means that you require *x* GB of memory per process.

7.8 Other PBS directives

There are several other #PBS directives one can use. Here a few of them are explained.

`-l walltime=`*`hh`*`:`*`mm`*`:`*`ss`*	Specify the maximum wallclock time for the job. After this time the job will be removed from the system.
`-l nodes=`*`n`*`:ppn=`*`m`*	Specify the number of nodes and cores per node to use. *n* is the number of nodes and *m* the number of cores per node. The total number of cores will be *n***m*
`-l mem=`*`x`*`mb`	Specify the amount of memory necessary for the job. The amount can be specified in mb (Megabytes), or gb (Gigabytes). In this case *x* Megabytes.
`-j oe`	Merge standard output and standard error of the jobs script in to the

	output file. (The option eo would combine the output into the error file).
`-e` *`filename`*	Name of the file where the standard error output of the job script will be written into.
`-o` *`filename`*	Name of the file where the standard output output of the job script will be written into.
`-m` *`events`*	Mail job information to the user for the given events, where *events* is a combination of letters. These letters can be: n (no mail), a (mail when the job is aborted), b (mail when the job is started), e (mail when the job is finished). By default mail is only sent when the job is aborted.
`-M` *`emails`*	e-mail adresses for e-mailing events. *emails* is a comma separated list of e-mail adresses.
`-q` *`queue_name`*	Submit to the queue given by *queue_name.*
`-S` *`shell`*	Change the interpreter for the job to *shell*.